**Sparse Matrix**

If you have worked with natural language uses cases, you would have definitely come across the process of converting text into the format that machines understand, i.e. numeric. While we are able to put it into the numeric format, we observe lot of zeros(majority) programmatically where it has ‘m’ rows and ‘n’ columns. Wondering where are we heading to, it is a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) in which most of the elements are zero.

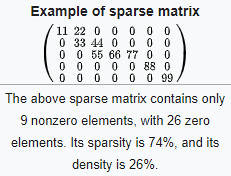
**Definition:**

Sparse Matrix/Sparse Array**:** A matrix is a two-dimensional data object made of m rows and n columns, therefore having a total m x n values. If most of the elements of the matrix have 0 value, then it is called a sparse matrix.

The number of zero-valued elements divided by the total number of elements (e.g., m × n for an m × n matrix) is called the sparsity of the matrix (which is equal to = 1 - the density of the matrix). Using those definitions, a matrix will be sparse when its sparsity is greater than 0.5.

Sparsity =

**If ( sparsity > 0.5 )** then : matrix is sparse **Else :** Not sparse .



Let’s quickly looks at the math:

Total elements: 35 , Non zero: 9 and Zeros: 26

and hence makes the sparsity= 26/35 and Density=9/35

**History:**

The term sparse matrix was possibly coined by Harry Markowitz

**Why learn Sparse matrix?**

* Analysing the book page by page, each word will be converted into a numeric format and considering 100000 words, there will be 100 words if we go page wise and results in a sparse matrix.
* Analysing the product review in e-commerce site, 1 product out of huge list will also boil down to sparse matrix when we look at holistically.
* Identifying the core topics in the documents.
* Summarising the large paragraphs.
* Sentiment analysis of tweets on social media.

Scenarios go on and if you are thinking to be an analyst that makes it implicit to learn the best use of the sparse matrix. Most importantly computers can understand only numbers and the above scenarios clearly result in the sparse matrix while converting text into Numbers.

I will take a simple example to show the sparse matrix generation in the context of text analytics:

We are analysing the sentences of a novel and below are the 2 sentences:

1. Mary had a little lamb
2. Little things really matter in life

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Words | Mary | had | a | little | lamb | little | things | really | matter | in | life |
| Sent1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Sent2 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

Imagine this for a large number of sentences and how sparse the words will be. It is evident we have to deal with it. Please note there are many ways of representing text as numbers (One hot encoding, Count Vectorizer Encoding and TF-IDF encoding)

**Challenges of Handling Sparse Matrix:**

* Memory: Even though the majority of elements are zero and has no information, space is utilized to store it and needs a way to handle it.
* Computational: To perform any calculations like matrix multiplication might take a lot of time due to although we know the result is 0. It has to still perform the process and time is wasted.

**Efficient Ways of Handling Sparse Matrix:**

* Storage:There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.
* Computing time**:** Computing time can be saved by logically designing a data structure traversing only non-zero elements.

When storing and manipulating sparse matrices on a computer, it is beneficial and often necessary to use specialized algorithms and data structures that take advantage of the sparse structure of the matrix.

Sparse data is by nature more easily compressed and thus requires significantly less storage.

Storing the Sparse Matrix: Format can be broadly classified into:

**Efficient Modification formats**

* Dictionary of keys (DOK): DOK consists of a dictionary that maps (row, column)-pairs to the value of the elements.
* List of lists (LIL): LIL stores one list per row, with each entry containing the column index and the value.
* Coordinate list (COO): stores a list of (row, column, value) tuples.

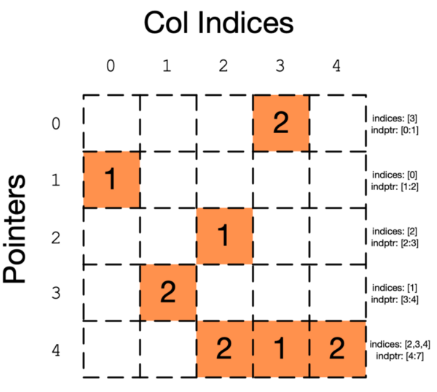
**Efficient Access and operations format**

* CSR (Compressed Sparse Row):

**data** is a row-wise flattened array

**indices** are indices of the columns

**indptr** is a pointer for data and indices, an array of length row + 1, with the max of its element as the length of the data or indices.

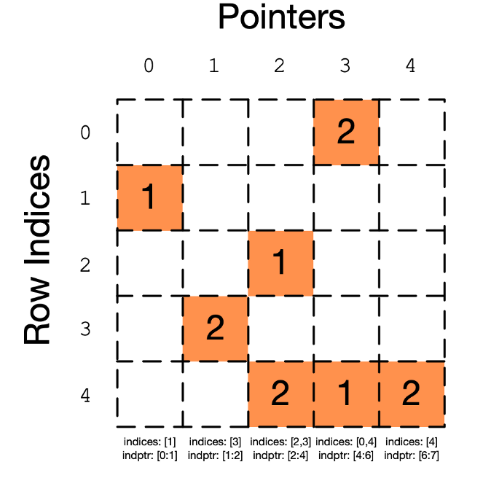


* CSC (Compressed Sparse Column)

**data** is simply a column-wise flattened version of the matrix

**indices** is the row indices for the corresponding elements of the data, e.g., the first element in the data is a 1, and it is located in the row index 1(second row); the second element in the data is a 2, and the row index is 3(fourth row), etc…

**indptr** is a pointer for data and indices, an array of length col + 1, with max of its element as the length of the data or indices. It is representing the range of column indices that each element in the data belongs to. In the above example, we have



**Conclusion**

In nutshell, Sparse matrix is common in most of the scenarios of natural language processing. Choosing the applicable compressing technique and efficiently utilising memory and computational power is key while analysing the unstructured data.